A Scrutiny of Actual Clothing Sizes in a Size Measurement Recommendation Model

PART 1

Abstract

It is essential for clothing brands to be able to recommend correct product sizes to their customer base to effectively increase sales as well as retain customers. Having proper size recommendations will reduce the amount of returned clothing, and therefore allow for businesses to provide much better customer experiences. In this paper, we use a dataset obtained from RentTheRunway clothing corporation to build a pipeline that uses a TF-IDF vectorizer and multiple classifiers to approach the product fitting problem and determine how the content of a customer's review can help us determine whether or not the product had a proper fit.

Introduction

Trade and commerce has been a long trait for humans throughout history. In a growing age where technology is deeply ingrained into our daily habits and lives, E-commerce, also known as electronic commerce, has been a popular choice for most people in our society. Each and every one of us are becoming significantly more busy as workloads continue to increase, therefore, the use of an online platform that would allow for users to save the trip of entering a store allows for enhanced convenience. However, this is where the problems arise. Perhaps the major concern with such a claimed convenience is how customers would recognize which sizes fit more than others. For many apparel pieces from a variety of different manufacturers signifies that there is also a large network of inconsistencies, variety, and differences. How would customers interested in renting/purchasing clothing pieces from a site determine which size may fit them without trying those physical pieces on first? In this project, we set out to determine whether or not a size fitting recommendation model could be set in place to prevent the need for customers to be stuck in the loop of having to obtain pieces and try them on to determine if they really do fit their unique bodies.

Dataset Description

Our recommendation model utilizes a dataset offered from the RentTheRunway clothing corporation, which is an uprising clothing rental service. This service serves the purpose of allowing interested individuals to rent clothing for certain occasions without the obligation of purchasing a high price tag item to wear only for a specific occasion. This obtained dataset contains 31MB of data specifically for reviews offered by customers after they rent clothing. Looking through the dataset, there is vast information offering features containing specific customer user data, individual product data, customer reviews, and their final voice on the fitting of the product.

Through exploring the dataset that we obtained, it was offered in a JSON format, javascript object notation. Since our project focuses on integrating with Python, our initial step in converting our data was represented as a dictionary in python terminology. Each individual dictionary represented one individual product review made by one individual user. An example dictionary of a review is below. Each dictionary roughly followed the following structure:

```
{'fit': 'fit',
 'user_id': '420272',
 'bust size': '34d',
 'item_id': '2260466',
 'weight': '137lbs',
 'rating': '10',
 'rented for': 'vacation',
 'review_text': "An adorable romper! Belt and
use, but that's to be expected. Wish it had pc
liments.",
 'body type': 'hourglass',
 'review_summary': 'So many compliments!',
 'category': 'romper',
 'height': '5\' 8"',
 'size': 14,
 'age': '28',
 'review_date': 'April 20, 2016'}
```

After observing the data that we retrieved, we wondered about the frequency of popular labels, categories, and values. As we looked through the variety of feature columns offered in the dataset, and conducted exploratory data analysis, these were our general dataset statistics:

Statistic	**Values**
Count of Transactions	192544
Count of Customers	105571
Count of Products	5850
Percentage of 'Small'	0.133886280538474
Percentage of 'Large'	0.128318721954462
Percentage of 'Fit'	0.737794997507063

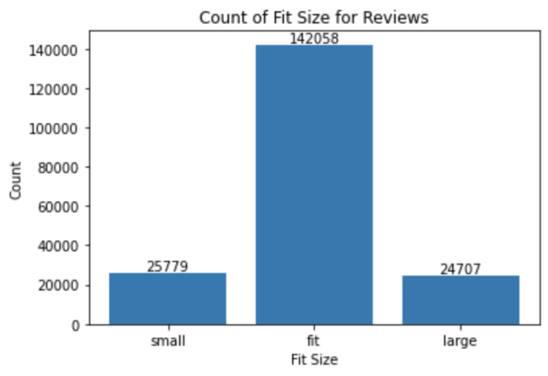

Figure 1: Count distribution for total reviews fit size. Majority are sizes that fit, but small and large sizes are roughly equal.

One specific interesting finding, as you can see from our visualization, is that amongst the reviews of our dataset, a disproportionately large number of reviews claimed that the item fit. Compared to the percentages of items that were too large or too small, an overwhelmingly large amount of items fit perfectly. This observation helped in the process of deciding which predictive task to focus our model on.

PART 2

Predictive Task

Among our analyses on this dataset, we decided to use the review text and the review summary of each individual review to predict whether the particular item of that review will be reported by customers as being too large, be too small, or fit perfectly. To begin our prediction task, we split the dataset into training, testing, and validation data subgroups. We then compiled our prediction model (created using the training data) by comparing our model's predictions for the testing and validation features to the actual outcomes within the testing and validation portions of the dataset. To determine the effectiveness of our prediction models, we followed up by calculating the separate testing and validation accuracies. These testing and validation accuracy values are used as a measure of the validity of our model's predictions. Our primary features of interest are the review text and review summary among our dataset. To obtain and organize these specific values, we created two empty lists, one for the review_text and one for review_summary feature columns. Following this, we looped over each individual review, extracted the review text and summary, converted the entire text and summary to lowercase letters only, removed all punctuation, and appended them to their corresponding list. This gave us the opportunity to work with individual lists of all of the review texts and all of the review summaries, rather than continue incorporating the dense python dictionary of values. By doing this, these lists further support the compilation process while building our prediction model by presenting the data in an accessible format. Our prediction model can be compared to the baseline model of using the presence of the words “small”, “fit”, or “large” in the review text to determine each customer’s classification on whether a product fits.

PART 3

Our Model

For our model, we first split our data into a training set consisting of 80% of the dataset, a testing set consisting of 10% of the dataset, and a validation set which accounted for 10% of the dataset. Our first step was to initialize a TfidfVectorizer instance object from Sci-Kit Learn, a popular machine learning package used among python, to fit and transform our lists obtained in previous steps that contained all review texts and review summaries. To do so, we used TfidfVectorizer’s fit_transform method, which learns the vocabulary and the idf (inverse document frequency) of all the documents and returns a document-term matrix all in one function. For our dataset, each review text and each review summary represented an individual document. Following this, we created a pipeline that listed out the steps that our model should take. Specifically, our pipeline would first instantiate the TfidfVectorizer instance, and a Sci-Kit Learn classifier. We used three different classifiers to determine which would result in the best accuracy for our model. The three different classifiers we used were SVC, Bagging Classifier, and K Neighbors Classifier.
SVC, also known as Support Vector Classification, works by taking in data points and mapping these respective data points on a graph to find the optimal feature value to use as the division between the three different prediction categories.
Bagging Classifier works by creating random subsets from the input data and fits those base classifiers on each of the subsets to aggregate predictions for each individual subset. The upside of this classifier is that it reduces variance due to its use of randomization. This randomization technique would create groups that don’t follow a specific order, but is instead random. The downside of this classifier is that it takes a significant period of time to run and fit. This is due to the backend splitting of groups and fitting of the model over many different subsets it splits on. For this reason, we had to limit our prediction training dataset when we were working with the review text. For our training on this model, we only used 20,000 data points on the review text feature columns to shorten the runtime of the model. However, we only had to shorten the dataset for the review text, but not the review summary. The review summaries were individually short enough that we were able to use the complete dataset, although it did take a while to run.
KNeighborsClassifier works by using the average of the outcomes of the K nearest neighbors of each datapoint to make a prediction about the classification.
After creating three different pipelines using TfidfVectorizer and each of the three aforementioned classifiers, we fit the X train and Y train values into three respective pipelines. These 3 different pipelines allowed us to be able to run the models efficiently with concise steps. We then used the X testing data and the X validation data to predict the test predictions and validation predictions. After building all 3 models, we calculated the accuracy of the outcome predictions obtained from each model by comparing them to the real classifications from the Y testing data and Y validation data. Our final accuracies returned consisted of a total of 6 different accuracies. This is motivated by the fact that we used TfidfVectorizer among 3 different classifiers (SVC, BaggingClassifier,

KNeighborsClassifier) and also calculated both the testing and validation accuracies among the variety of classifiers that we tested. Using both the testing and validation accuracies of all three pipelines, we were able to determine the best classifier. The reason we decided to use this model is because we know that the TF-IDF vectorizer is one of the best models to use for large amounts of text, since the metric of TF-IDF is a good interpreter of the importance of words in a document. A model that we tried, but was unsuccessful, is LinearRegression. We were unable to obtain realistic results, because we kept getting decimals and almost 0% accuracy. This potentially could have occurred because we were using the review text to predict the rating. The length of review texts and the ratings were very inconsistent and could have led to this issue. However, the TF-IDF method worked well with high accuracy and minimal overfitting.

PART 4

Relevant Literature

While researching for relevant literature, multiple papers have referred to and discussed models pertaining to fitting recommendations for clothing, but the concept itself is somewhat recent. Compared to other research in this field, our approach takes a different route on the challenge of determining fitting recommendation among clothing. Most research focuses itself on recommending sizes for its users, but rather we are trying to determine whether or not a user was able to obtain the right size based on their review. Among current research which similarly utilizes the RentTheRunway dataset, all has been primarily involved in developing a new predictive framework for the product fitting problem that we are also trying to tackle [1]. We got this data by directly downloading it off the UCSD CSE158 course webpage, but other researchers are also able to obtain the data from publicly available datasets from online clothing retailers. In RentTheRunway's research, this dataset was used to build a product size recommender system for customers to have a better understanding of the correct sizes they should obtain for their products. The researchers' approach was to focus on capturing fit semantics and be able to handle label imbalance issues through metric learning approaches that involve prototyping. Using this data, this study advocates for using a more effective heuristic to use for choosing the optimal prototype, called the Large Margin Nearest Neighbor (LMNN) [1].

Other studies have also been done to tackle this same problem but with different, yet similar datasets. One approach to the fitting problem looks at a customer's true sizes and uses them as a feature in a classifier [3]. Another study extends that approach and also offers another approach of using Bayesian logit and probit regression models that have ordinal categories to use for modeling [4]. A more recent study uses skip-gram models to learn of the latent features of the customers and products [2].

Overall, our own prediction findings contrast those from other studies, due to the fact that we employ a significantly different approach than all surrounding studies. Not only that, we try to answer a completely different aspect of the product fitting problem. The classification techniques used in these studies are a bit more complex than we have learned, so we tried to use a more simpler model through TF-IDF. In our research, we simply tried to predict whether a customer obtained a proper fit based on the content of their review that they gave after receiving the product. We used a pipeline with the TF-IDF vectorizer and other classification methods to try and create our own model. It is important to note that our work would not be able to replace the work done in all of these studies, but instead could be used in conjunction with existing research to continue the challenge of predicting how well the current models in place are performing and work in the real world.

<u>PART 5</u>

Results and Analysis

Feature	Classifier	Testing Accuracy	Validation Accuracy
Review text	SVC	0.8028564	0.8056608
	BaggingClassifier (only 20000 datapoints)	0.764	0.782
	KNeighborsClassifier	0.7359127	0.7410542
Review Summary	SVC	0.7586081	0.7544014
	BaggingClassifier	0.7371591	0.7341989
	KNeighborsClassifier	0.7345624	0.7352376

Our results are summarized in the table above. Overall, we have two different approaches in trying to solve the fitting problem that we are trying to produce a solution for. Both approaches involve the usage of the TF-IDF vectorizer and create an implementation of a classifier within it inside of a pipeline. In one approach, we created a model that allows us to use the review text feature to determine whether a product had a proper fit for the customer. In the other approach, we used the review summary feature in the data to also determine whether the product in question would have a proper fit on the customer.

We implemented three classifiers within both approaches to try and obtain the best results. Overall, the Linear SVC classification method that we added into the pipeline along with the TF-IDF vectorizer yields the best results for the review text approach. With the Review Text feature, we obtained a testing accuracy of around 80%. Along with this classifier we also implemented the Bagging classifier, which yields around 76% testing accuracy and the KNearest Neighbors classifier which yields around 73.6% accuracy. Our model does not overfit the data because our validation accuracy is also very close to the testing accuracy. For the Review Summary approach, we use the same classifiers, but are unable to obtain as high of an accuracy as the Review Text approach. Again, the Linear SVC classifier gives the highest testing accuracy for the review summary as well, at around 75.9%, while the Bagging classifier and KNeighbors classifiers produce testing accuracies of around 73.7% and 73.5% respectively.

In both feature approaches, the Linear SVC classifier clearly outperforms the Bagging Classifier and the KNeighbors Classifer. They all are able to produce decent accuracy results, but with the Review Text feature with the Linear SVC classifier produce the best results. None of the models failed necessarily, but we do believe that the Bagging Classifier would perform better if it had more data points. We did have to cut the total data points in our study to ensure that our kernel would actually run. With better technology, there is definitely a potential for this classifier to perform much better.

Conclusion

In this paper, we describe a method to determine whether a customer received a correctly fitting product. To help us achieve this, we use the RentTheRunway dataset and build a pipeline involving a TF-IDF vectorizer and 3 different classifiers to determine which gave us results most similar to those that we were given in the dataset. Through using TF-IDF alongside Support Vector Classification, Bagging Classification, and KNeighbors Classification models, we concluded that the Linear Support Vector Classification yielded the best results, therefore, is our most accurate model.

By creating such a model that generates a classification output on whether or not a specific product fits or not, we would be able to take a step further in ensuring great customer services through the E-commerce market. There is a lot of scope for further research and studies in this domain.

References

[1] Rishabh Misra, Mengting Wan, and Julian McAuley. 2018. Decomposing fit semantics for product size recommendation in metric spaces. In Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18). Association for Computing Machinery, New York, NY, USA, 422–426.

[2] G Mohammed Abdulla and Sumit Borar. 2017. Size Recommendation System for Fashion E-commerce. KDD Workshop on Machine Learning Meets Fashion (2017).

[3] Vivek Sembium, Rajeev Rastogi, Atul Saroop, and Srujana Merugu. 2017. Recommending Product Sizes to Customers. In RecSys.

[4] Vivek Sembium, Rajeev Rastogi, Lavanya Tekumalla, and Atul Saroop. 2018. Bayesian Models for Product Size Recommendations. In Proceedings of the 2018 World Wide Web Conference (WWW '18). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 679–6